Journal of Reliable Intelligent Environments (2020) 6:79–94
https://doi.org/10.1007/s40860-020-00098-y

ORIGINAL ARTICLE

A secure mutual authentication protocol for IoT environment

Prabhat Kumar Panda[1] · **Sudipta Chattopadhyay**[2]

Received: 18 August 2019 / Accepted: 4 January 2020 / Published online: 16 January 2020

Abstract
Rapid development in the field of Internet of Things (IoT) has made it possible to connect many embedded devices to the internet for the sharing of data. Since, the embedded device has limited storage, power, and computational ability, an integration of embedded devices with the large pool of resource such as cloud is required. This integration of technologies is expected to provide extraordinary growth in current and future promising applications of IoT. In this context, the security issues such as authentication and data privacy of devices are major issues of concern. The research motivation of the present work is to propose a secure mutual authentication protocol for IoT and cloud servers based on elliptic curve cryptography. In this work, the security properties of the proposed protocol have been formally verified by using Automated Validation of Internet Security Protocols and Applications tools and informally analyzed and compared with the related protocols in terms of various security attributes such as device privacy, impersonation attack, replay attack, password guessing attack, mutual authentication and so on. Moreover, the performance of the proposed protocol has also been evaluated in terms of computational, communication, storage overhead and total computational time. The security and performance analyses found the supremacy of the proposed protocol over the other related protocols.

Keywords Authentication · Cloud server · Elliptic curve cryptography · Internet of Things · Security

1 Introduction

Internet of Things (IoT) [1] is a network of physical devices, objects, buildings, vehicles and other things that are embedded with software, electronics, sensors, and network connectivity. These objects are connected together and interchange the information between them and with other digital devices without any human interference [2, 3]. IoT contributes to boosting the life we live in through many applications such as smart cities, e-healthcare, smart buildings, smart grids and many more.

In recent years, due to the rapid development of IoT, internet connectivity with embedded devices for information sharing has also increased. Since the embedded device has limited storage, power, and computational ability, it is integrated with the cloud server, where the cloud has more storage and processing power and also can resolve most of the IoT issues. Combining the IoT devices with the cloud makes a new paradigm named CloudIoT which is expected to provide an extraordinary growth in current and future internet [4]. In the CloudIoT environment, the embedded device can depend on the computational skill of the cloud and can extract a large amount of data storage from the cloud server. Moreover, the embedded devices are more suitable for the practical implementation of IoT which results in different types of IoT services by incorporating smart embedded devices. However, while connecting an embedded device to a cloud, security is the prime issue of concern [5, 6]. Also, the mutual authentication must be established between the cloud server and the embedded devices. To meet these security requirements, many authentication protocols have been proposed for IoT and cloud servers. However, the existing protocols have certain shortcomings which need to be addressed further. In the environment, where memory and power are limited and higher security needs to be achieved at a minimum key length, then elliptic curve cryptography

✉ Prabhat Kumar Panda
prabhatjdvu@gmail.com

Sudipta Chattopadhyay
sudiptachat@yahoo.com

[1] School of Electronics and Communication Engineering, REVA University, Bangalore 560064, India

[2] Department of Electronics and Telecommunication Engineering, Jadavpur University, Kolkata 700032, India

(ECC) is considered to be the best public key cryptography scheme [7].

Being motivated by the above research issues and trends, an improved mutual authentication and security protocol for IoT environments based on ECC has been proposed in this paper.

The major contributions of this work are summarized below:

- The ECC technique has been adopted to eliminate several security issues.
- The proposed protocol employs the concept of password verifier with the status bit in such a way that the server stores the password in the form of a password verifier with a status bit to achieve the device privacy and to prevent the impersonation attack and many logged-in devices' attack.
- Proper mutual authentication and perfect forward secrecy have been achieved by following a unique way of computing the values of several authentication parameters and session key.
- The formal security verification of the proposed protocol by using the Automated Validation of Internet Security Protocols and Applications (AVISPA) tool has been provided.
- An informal security analysis of the proposed protocol has also been carried out with respect to several security attributes such as mutual authentication, device privacy, impersonation attack, replay attack, offline password guessing attack, many logged-in device attacks, insider attack, session key agreement, perfect forward secrecy, etc., and compared with the existing protocols to establish the supremacy of our work over the existing ones.
- The performance analysis of the proposed work has been compared with the existing work for computational overhead, communication overhead, storage overhead and total computational time. The results of the analysis show that the proposed protocol outperforms the related work in this regard.

The remainder of this paper has been structured as follows: In Sect. 2, related work to the proposed protocol has been described. In Sect. 3, preliminaries of the ECC have been summarized. Section 4 describes the methodology of the proposed protocol. Formal and informal security analysis of the proposed protocol has been analyzed and compared with the related protocol with respect to several security attributes in Sect. 5. In Sect. 6, the performance of the proposed protocol has been analyzed and compared with the related protocols to different performance parameters. Finally, some concluding notes and outline for future directions have been included in Sect. 7.

2 Related work

Authentication plays an important role for the successful integration of embedded devices with cloud computing services. Recently, several authentication protocols have been proposed for smart devices. Many authentication protocols based on ECC which apply to smart devices have been proposed in [8–17]. However, they have their own merits and demerits. The protocol proposed by Yang et al. [8] offers mutual authentication and also supports session key agreement between the user and the server. Afterward, Yoon et al. [9] analyzed that the protocol [8] does not offer perfect forward secrecy. Moreover, it gets affected by the impersonation attack. To overcome these issues, the author in [9] proposed an improved protocol to provide better security. Later, Islam et al. [10] found that the protocol [9] also fails to provide forward secrecy. Subsequently, the authors proposed a secure identity-based remote login protocol with a three-way challenge-response handshake technique. The protocol in [10] removes the clock synchronization problems, reduces the computational cost and also provides better security than the above protocols. In 2013, Chou et al. [11] analyzed the protocols [8, 9] and pointed out that users do not have the appropriate public key in the protocols [8, 9]. Moreover, in [11] the authors developed two ID-based key agreement protocols for mobile environments based on ECC. Next, in [12], Farash et al. reviewed the protocol [11] and found that the protocol [11] is vulnerable to impersonation attack. To overcome the limitations, the author proposed an enhanced ID-based key exchange protocol. However, the computational cost of the protocol [12] is higher than that of the protocol in [11].

Liao et al. [13] proposed an RFID authentication protocol combined with the ID-verifier transfer scheme. The authors claimed that their protocol offers mutual authentication and resist various security attacks. However, Peeters et al. [14] showed that the protocol [13] does not achieve mutual authentication and privacy. Moreover, it also gets affected by server spoofing attack. In 2014, Moosavi et al. [15] developed a mutual authentication protocol for RFID system based on ECC. The authors demanded that their protocol is immune to several attacks. However, Khatwani et al. [16] analyzed the protocol [15] and proved that the protocol [15] is affected by a kind of denial-of-service (DoS) attack, i.e., clogging attack. Abbasinezhad-Mood et al. [17] proposed a novel ECC-based self-certified two-factor key management scheme for medical data protection. The authors have been used ProVerif tool to proof the security features of their proposed scheme. Moreover, to compute the execution time, they have implemented the cryptographic elements on hardware's. Some other authentication and key establishment protocols developed in [18, 19] which have been proofed the security features of the protocol by using ProVerif tool. The efficiency of the both protocols [18, 19] has been evaluated experimentally by using Advanced RISC Machines (ARM) platforms.

Meanwhile, many authentication protocols for the IoT environment have also been proposed. A secure lightweight mutual authentication protocol for IoT smart home has been proposed by Alshahrani et al. [20] based on cumulative keyed hash chain. The authors adopt cumulative keyed hash chain to confirm the identity of the sender. In this protocol, Automated Validation of Internet Security Protocols and Applications (AVISPA) and Burrows-Abadi-Needham (BAN) logic have been used to validate the security of the protocol. An ECC-based secure authentication protocol with privacy protection for Industrial Internet of Things (IIoT) has been developed by Li et al. [21]. The authors presented a biometric-based authentication with ECC to mitigate the security flaws. The security of this work has been proved under random oracle model. Moreover, the work has been simulated by using NS-3 and the authors claimed that the protocol is more suitable for IIoT environment. Alcaide et al. [22] established a decentralized anonymous authentication scheme for the users in the IoT environment. The scheme holds some exponentiation operations and is suitable for powerful platforms. Nevertheless, Lin et al. [23] pointed out that the adversary can capture the data from data collectors by impersonating the user. In 2017, a remote-user authentication protocol by using three factors such as passwords, smart cards, and biometrics for IoT environments was proposed by Dhillion et al. [24]. This protocol only uses hash and XOR operations which are appropriate for the resource-constrained nodes and devices. The authors proved that it is resistant to many security attributes such as DoS attack, impersonation attack, stolen smart device attack, and offline password guessing. To mitigate the security flaws which are shown in several light weight two-factor or three-factor authentication and key agreement protocols, Ostad-Sharif et al. [25] proposed an three-factor authentication and key agreement protocol for IoT-based Wireless Sensor Network. The formal security analysis of this protocol has been validated by using AVISPA tool. The authors claimed that this work is efficient and appropriate for IoT-based WSN environments.

Based on dynamic reconstruction of metadata, a structure for preservation of cloud users' data privacy has been established by Waqar et al. [26]. The authors also used the mechanisms of database table splitting, data classification, and data encryption/decryption for protecting the metadata stored in cloud's database. A top-down utility paradigm for cloud and IoT by using mobile devices and sensor networks has been established by Distefano et al. [27]. To achieve efficient communication between the device and cloud, a framework for integrating the IoT and cloud in a unified programming model has been proposed by Persson et al. [28]. Stergiou et al. [29] presented a survey on IoT and cloud computing by focusing on the security issues of these technologies. Moreover, the authors integrated both technologies to determine the common features and to examine the benefits of the combination. Furthermore, the authors proposed an algorithm to survey the security challenges of the merged IoT and cloud computing. An authentication scheme was developed by Chatterjee et al. [30] which uses three-way approaches for IoT environment based on ECC. The authors perform the security analysis and claims that their protocol secure against various cryptographic attack.

Another ECC-based authentication protocol for IoT and cloud environments has been developed by Kalra et al. [31]. The authors claimed that their protocol offers mutual authentication using the HyperText Transfer Protocol (HTTP) cookies. Additionally, they proved that the protocol is resistant to several security attacks. However, Chang et al. [32] found that the protocol in [31] failed to achieve mutual authentication and the session key agreement is infeasible. The authors also tried to overcome the security flaws of the protocol [31] by establishing an improved authentication protocol for IoT and cloud environments. Afterward, in 2017, Wang et al. [33] reviewed the protocols [31, 32] and pointed out that both of the protocols [31, 32] are insecure. Subsequently, the authors proposed a secure authentication protocol for IoT networks and ensured the security of their protocol. However, the protocol in [33] failed to achieve device privacy and vulnerable to impersonation attack and many logged-in devices' attack. Kumari et al. [34] analyzed and found that the protocol [31] does not offer mutual authentication, affected by various security attacks and session key agreement is infeasible. To overcome these security flaws, the authors proposed an improved authentication protocol for IoT environment based on ECC. However, this protocol consumes more computational cost and storage cost as compared to the protocol [31]. In 2018, Bhubaneswari et al. [35] also analyzed the protocol [31] and showed that the protocol is vulnerable to several security attacks and subsequently approached an enhanced mutual authentication protocol for IoT network. However, this protocol does not provide mutual authentication and also unable to offer perfect forward secrecy.

The advantages and disadvantages of the most relevant authentication schemes to the proposed protocol are summarized in Table 1.

3 Preliminaries

3.1 Elliptic curve cryptography (ECC)

Elliptic curve cryptography is a public key cryptography technique which depends on the algebraic structure of

Table 1 Analysis of relevant authentication protocols

Literature	Authentication scheme	Advantages	Disadvantages
Hafizul et al. [10]	An efficient and secure ID-based remote mutual authentication with key agreement scheme for mobile devices on elliptic curve crypto systems	Removes the clock synchronization problems Reduces the computational cost Resistant to replay, insider, impersonation and many logged-in device's attacks Provides perfect forward secrecy and achieves mutual authentication	Computational overhead is little high
Liao et al. [13]	A secure ECC-based RFID authentication scheme integrated with ID-verifier transfer protocol	Secure against replay and many logged-in device's attacks Provides perfect forward secrecy	Does not achieves Mutual authentication Affected by server spoofing attack
Kalra et al. [31]	Secure authentication scheme for IOT and cloud servers	Resistant to replay attack	Failed to achieve mutual authentication Absence of device anonymity
Chang et al. [32]	Notes on secure authentication scheme for IOT and cloud servers	Secure against replay attack Achieves mutual authentication Provides perfect forward secrecy	Vulnerable to password guessing, impersonation, insider and many logged-in devices attack Absence of device anonymity
Wang et al. [33]	A secure authentication scheme for internet of things	Resistant to replay attack. Achieves mutual authentication Provides perfect forward secrecy	Failed to achieve device privacy Affected by impersonation, and many logged-in devices attack
Kumari et al. [34]	A secure authentication scheme based on elliptic curve cryptography for IoT and cloud servers	Secure against replay, password guessing attack and insider attack Provides perfect forward secrecy and achieves mutual authentication Achieves device privacy	Vulnerable to impersonation, and many logged-in devices attack
Bhubaneswari et al. [35]	Enhanced mutual authentication scheme for cloud of things	Resistant to replay attack, password guessing attack and insider attack Achieves device privacy	Does not achieve Mutual authentication Affected by impersonation, and many logged-in devices attack Fails to provide perfect forward secrecy

elliptic curves over finite fields Z_q [36]. Most of the current cryptographic systems prefer to use ECC to achieve greater security and efficient computation. The security strength of the ECC mainly lies in the difficulty involved to solve the elliptic curve discrete logarithm problem (ECDLP). It can provide an equivalent level of security as of RSA by using fewer key bits [36], i.e., the 160-bit elliptic curve key achieves the equivalent level of security strength as RSA key size of 1024 bits [37]. A brief overview of ECC is analyzed below:

The equation of the elliptic curve $E_q(a, b)$ over Z_q is written as $y^2 \bmod q = x^3 + ax + b(\bmod q)$, where q is a large prime number and a and b are two constant $(a, b \in Z_q)$ such that the condition $4a^3 + 27b^2 \neq 0$ should be satisfied. Any point $(x, y) \in E_q(a, b), x, y \in Z_q$ together with O forms an additive cyclic group $E_g = \{(x, y) \in E_q(a, b)\} \cup \{O\}$, where O is defined as 'point at infinity.' The point multiplication on the cyclic group is computed by repeated addition, i.e., $m \cdot P = \overbrace{P + P \cdot \cdots + P}^{m\ times}$. The further details of the elliptic curve cryptosystem properties are analyzed in [36].

The computational problems over E_g have been described below [36, 38, 39]:

Definition 1 (*ECDLP: Elliptic Curve Discrete Logarithm Problem*): Given P, $Q \in E_g$, difficult to find an integer $m \in [1, n-1]$, such that $Q = m \cdot P$.

Definition 2 (*CDHP: Computational Diffie Hellman Problem*): For $a, b \in [1, n-1]$, given P, aP and bP, difficult to compute abP.

4 The proposed protocol

In this section, a secure authentication protocol based on ECC has been proposed for the IoT environment. Here, various phases of the proposed protocol have also been described. The notations which are used in the proposed protocol are listed in Table 2.

The operational workflow diagram of the proposed protocol is presented in Fig. 1. The proposed protocol consists

Table 2 Notations used in the proposed protocol

Notations	Descriptions
ED_i	An embedded device
CS	The cloud server
ID_i	The identity of the device ED_i
E	An elliptic curve equation
$E_q(a, b)$	An elliptic curve, where a and b are two constant
E_g	An elliptic curve group over E
P	Public point/generator point of the elliptic curve group with order n such that $n \cdot P = 0$
q, n	Large prime numbers
Z_q	A finite field over a large prime number q
PW_i	Password of device ED_i
PV_i	Password verifier of device ED_i, where $PV_i = PW_i \cdot P$
X_{CS}	Server's secret key select from $[1, n-1]$
R_S	Server's random number
R_1, R_2	Random numbers select from $[1, n-1]$
$H()$	One-way cryptographic hash function
CK	Cookie information
E_T	Expiration time of the Cookie
SK	Session key individually generated by ED_i and CS

of two phases: (1) Registration phase and (2) Login and authentication phase. These phases are described as follows:

4.1 Registration phase

Step 1 ($ED_i \rightarrow CS$): ED_i

1. At the initial stage of the network entry, to register with the cloud server CS, the embedded device ED_i computes protected identity $I_i = H(ID_i)$ and generates a unique password PW_i for each device ED_i. Then, it computes the password verifier $PV_i = PW_i \cdot P$ and sends $\{I_i, PV_i\}$ to CS, where password verifier PV_i has been computed and sends to achieve the device privacy and to prevent the impersonation attack and many logged-in devices' attack.

Step 2 ($CS \rightarrow ED_i$): CS

1. After receiving the registration request, CS stores $\{I_i, PV_i\}$ and a status bit into a write protected mode as defined in Table 3. Here, the status bit signifies the current status of the device, i.e., when the device is logged into the server, the status bit is set to one '1,' otherwise it is set to zero '0.'
2. Generates a random number R_S and computes the cookie CK,

$$CK = H(R_S \parallel X_{CS} \parallel E_T \parallel I_i)$$

$$CK' = CK \cdot P.$$

3. Calculates the other security parameters as follows:

$$T_i = R_S \oplus H(X_{CS})$$

$$A_i = H(R_S \oplus H(X_{CS}) \oplus CK')$$

$$A_i' = A_i \cdot P.$$

4. Stores $\{ET_i = T_i \oplus X_{CS}, EA_i' = A_i' \oplus R_S$ and $EE_T = E_T \oplus R_S\}$ corresponding to I_i of the device ED_i in its database. Here, the security parameters are encrypted and then stored to avoid the impersonation attack.
5. Afterward, CS sends CK' to the embedded device ED_i in a secure channel.

Step 3 ED_i

1. After receiving CK', the embedded device stores CK' in its memory.

4.2 Login and authentication phase

Step 1 ($ED_i \rightarrow CS$): ED_i

1. Before every login, it generates a random nonce R_1 and then calculates the values of P_1,P_2 using the formulas:

$$P_1 = R_1 \cdot PW_i \cdot P$$

$$P_2 = H(R_1 \cdot PW_i \cdot CK')$$

2. It encrypts I_i such as, $EI_i = I_i \oplus K_{PV}$ where, $K_{PV} = PV_x \oplus PV_y$. Here, K_{PV} has been derived by performing the XOR of the ECC point (PV_x, PV_y) and used to encrypt the protected identity I_i.
3. Next, it sends the login request with $\{P_1, P_2, EI_i\}$ to the server.

Step 2 ($CS \rightarrow ED_i$): CS

1. After receiving the login request, it decrypts I_i by using K_{PV} and validates by checking I_i to know whether ED_i is a legal device or not. If not, rejects the login request. If yes, it retrieves the data associated with received I_i from its database. Then, calculate different parameters as follows:

$$T_i = ET_i \oplus X_{CS}$$

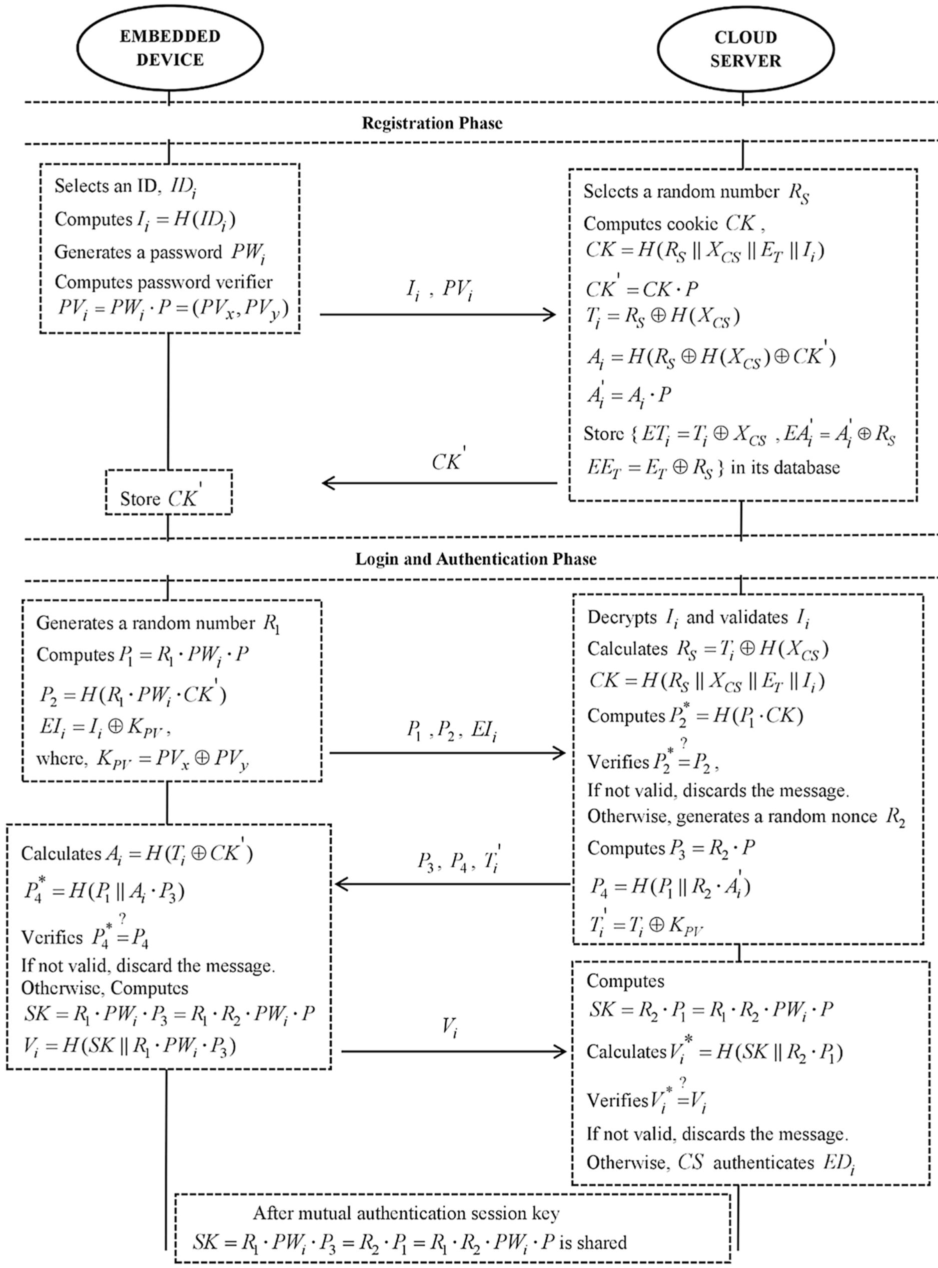

Fig. 1 The operational flow diagram of the proposed protocol

Table 3 The verifier table with device status bit

Device identity (protected)	Password verifier	Status bit
I_1	$PV_1 = PW_1 \cdot P$	0/1
I_2	$PV_2 = PW_2 \cdot P$	0/1
I_3	$PV_3 = PW_3 \cdot P$	0/1
–	–	–

$$R_S = T_i \oplus H(X_{CS})$$

$$E_T = EE_T \oplus R_S$$

$$A_i' = EA_i' \oplus R_S$$

$$CK = H(R_S \parallel X_{CS} \parallel E_T \parallel I_i).$$

2. Computes $P_2^* = H(P_1 \cdot CK)$ and verifies $P_2^* \stackrel{?}{=} P_2$.
3. If the above condition is not valid, it discards the message; otherwise, it generates a random nonce R_2 and computes the values of P_3, P_4 and T_i' as follows:

$$P_3 = R_2 \cdot P$$

$$P_4 = H(P_1 \parallel R_2 \cdot A_i')$$

$T_i^{'} = T_i \oplus K_{PV}$.

4. Afterward, CS sends { P_3,P_4,$T_i^{'}$ } to ED_i for authentication.

Step 3 ($ED_i \rightarrow CS$): ED_i

1. After receiving { $P_3, P_4, T_i^{'}$}, it calculates $T_i = T_i^{'} \oplus K_{PV}$ and then $A_i = H(T_i \oplus CK^{'})$.
2. Computes $P_4^* = H(P_1 \parallel A_i \cdot P_3)$ and verifies $P_4^* \stackrel{?}{=} P_4$.
3. If above condition is not satisfied, discard the message $\{P_3, P_4, T_i^{'}\}$; otherwise, ED_i authenticates CS and continues the process.
4. Afterward, it computes the session key $SK = R_1 \cdot PW_i \cdot P_3 = R_1 \cdot R_2 \cdot PW_i \cdot P$ and compute and sends a verifier $V_i = H(SK \parallel R_1 \cdot PW_i \cdot P_3)$ to CS for authentication.

Step 4 CS

1. After receiving the verifier V_i, it calculates the session key $SK = R_2 \cdot P_1 = R_1 \cdot R_2 \cdot PW_i \cdot P$.
2. Computes $V_i^* = H(SK \parallel R_2 \cdot P_1)$ and verifies $V_i^* \stackrel{?}{=} V_i$.
3. If the above condition is false, V_i is discarded. Else, CS authenticates ED_i to achieve mutual authentication.
4. After mutual authentication between ED_i and CS, the session key $SK = R_1 \cdot PW_i \cdot P_3 = R_2 \cdot P_1 = R_1 \cdot R_2 \cdot PW_i \cdot P$ is shared between them and all the consequent messages are transmitted between them by performing XOR operation with SK.

5 Security analysis

This section presents the attack model to show the capabilities of adversary, formal security verification using Automated Validation of Internet Security Protocols and Applications (AVISPA) tools to show the proposed protocol is secure against various attacks and also analyzes different security attributes related to the proposed protocol by the informal security analysis.

5.1 Attack model

Security is the most important part while designing the IoT model. In order to design attack free and more secure IoT devices and applications below issues should be addressed [24]:

- *Denial-of-Service attack* An adversary may disturb the network by overloading with the fake messages to degrade the performance of the network and making service unavailable. This will help the adversary to make the resources unavailable to the intended users.
- *Eavesdropping attack* The adversary may intercept the messages and read the ongoing communication between embedded device and cloud server. Subsequently, adversary may store the information and used that to launch the eavesdropping attack.
- *Password guessing attack* By using offline dictionary attack, an adversary can try to guess the password of the legal device to make feasible the attack.
- *Impersonation attack* By sending the valid messages of the previous communications with in the valid entities, an adversary can impersonate as a legal device.
- *Man-in-the-middle attack* At the time of live communication is going on with in two legitimate entities, an adversary can try to listen it. Later on, he can delete, alter or delay the transmission messages.

5.2 Formal security verification using AVISPA

The formal security verification of the proposed protocol through the simulation using the AVISPA [40, 41] tool has been performed. AVISPA is a push-button tool for automated validation of internet security protocols, which is a commonly accepted tool for formal security verification [42]. It integrates four back-ends: On-the-fly Model-Checker (OFMC), Constraint Logic-based Attack Searcher (CL-AtSe), SAT-based Model-Checker (SATMC) and Tree Automata based on Automatic Approximations for the Analysis of Security Protocols (TA4SP). The detailed analyses of these back-ends are described in [40]. The role oriented language such as High-Level Protocol Specification Language (HLPSL) [40] in AVISPA has been used for implementing the security protocols. This language contains the basic roles and composition roles representing each participant role and the scenarios of basic roles, respectively. An intruder (i) is modeled by using the Dolev–Yao model [43]. Consequently, in the protocol run time, the intruder (i) is permitted to act a legitimate role. In HLPSL, some basic roles, a number of principals and a number of sessions are defined. The HLPSL2IF translator is used to convert HLPSL to intermediate format (IF). The IF is then used as input to any one of the four back-ends which produces output format (OF). The detailed description of the OF is presented in [40].

The proposed protocol is simulated by using the Security Protocol Animator for AVISPA (SPAN) [40] under the OFMC and CL-AtSe back-ends. To check the chance of a replay attack, both the back-ends verify if the specified legitimate agents can execute the specified protocol by performing a search of a passive intruder. The back-ends provide the intruder (i) about the information of some normal sessions between the legitimate agents. Subsequently, both the back-ends also verify if there is any possibility of a man-in-the-middle attack by the intruder for the Dolev–Yao model checking. The simulation has been done to show the proposed protocol is secure and safe against various security attacks.

The HLPSL code developed for simulation is shown in Fig. 2, 3 and 4. The simulation results of the analysis under both back-ends are presented in Fig. 5. The simulation results ensure that the proposed protocol is safe from replay and man-in-the-middle attack.

5.3 Informal security analysis

This section analyzes different security attributes related to the proposed protocol and compares them with the other related protocols [10, 13, 31–35]. The result of the analysis is summarized in Table 4.

5.3.1 S1: mutual authentication

In the proposed protocol, during login and authentication process, cloud server authenticates embedded device by verifying $P_2^* \stackrel{?}{=} P_2$ and $V_i^* \stackrel{?}{=} V_i$. In step 1 of login and authentication phase, the device computes $P_2 = H(R_1 \cdot PW_i \cdot CK')$ which is only computed by a legal device and sends it to the cloud server. Then, the server computes $P_2^* = H(P_1 \cdot CK)$ where, $P_1 = R_1 \cdot PW_i \cdot P$ and verifies $P_2^* \stackrel{?}{=} P_2$. Similarly, in step 3 of the login and authentication phase, the device computes $V_i = H(SK \parallel R_1 \cdot PW_i \cdot P_3)$ and sends it to cloud server. Next, the cloud server verifies $V_i^* \stackrel{?}{=} V_i$ to authenticate embedded device. Also, the device authenticates the cloud server by verifying $P_4^* \stackrel{?}{=} P_4$. In step 2 of the login and authentication phase, the server computes $P_4 = H(P_1 \parallel R_2 \cdot A_i')$ where, $A_i' = A_i \cdot P$ and sends it to device. After this, the device computes $P_4^* = H(P_1 \parallel A_i \cdot P_3)$

Fig. 2 HLPSL code for role specification of Edi

```
role embedded_device (EDi, CS: agent,SK: symmetric_key, H: hash_func, SND, RCV: channel(dy))
played_by EDi
def=
      local State: nat,
      IDi, Ii, PWi, PVi, CK, CK1, Rs, XCS, Et: text,
      P, R1, R2, P1, P2, P3, P4, PVx, PVy, KPV, EIi, Ti, SK1, Vi: text,
      E: hash_func
      const s1, s2, ed_cs_r1, cs_ed_r2 : protocol_id
      init State := 0
      transition
      % Registration phase
                1. State= 0 /\ RCV (start) =|>
                  State':=1/\ IDi':=new ()
                            /\ PWi':=new ()
                            /\ Ii':=H (IDi)
                            /\ PVi':= E (PWi'. P)
                            /\ secret ({IDi, PWi}, s1, EDi)
                            /\ SND ({Ii'. PVi'} _SK)

                2. State=1 /\ RCV ({CK'} _SK) =|>
                  State':=3 /\ secret ({XCS, Et}, s2, EDi)
      % Login and Authentication phase
                            /\ R1':= new ()
                            /\ Ii':= new ()
                            /\ P1':= E (R1'.PWi.P)
                            /\ CK1':={{ CK'} _SK} _SK
                            /\ P2':= H (R1'.PWi.CK1')
                            /\ KPV':= xor(PVx, PVy)
                            /\ EIi':= xor(Ii', KPV')
                            /\ SND ({P1'. P2'. EIi'} _SK)
                            /\ witness (EDi, CS, ed_cs_r1, R1')

                3. State= 3 /\ RCV ({Ti'. P3'. P4'} _SK) =|>
                  State':= 5 /\ R1':= new ()
                            /\ R2':= new ()
                            /\ SK1':= E (R1' . E (R2'. E (PWi . P)))
                            /\ Vi':= H (SK1' . E (R1' . E (PWi . P3')))
                            /\ SND ({Vi'} _SK)
                            /\ request (CS, EDi, cs_ed_r2, R2')
end role
```

Fig. 3 HLPSL code for role specification of CS

```
role cloud_server (EDi, CS: agent, SK: symmetric_key, H: hash_func, SND, RCV: channel (dy))
Played_by CS
def=
    local State: nat,
    IDi, Ii, PWi, PVi, RS, CK, CK1: text,
    XCS, ET, P, R1, R2, P1, P2, P3, P4, Ai, Ai1, EIi, Ti, Ti1, PVx, PVy, KPV, Vi: text,
    E: hash_func
    const s1, s2, ed_cs_r1, cs_ed_r2: protocol_id
    init State := 0
    transition
    % Registration phase
            1. State=0 /\ RCV ({Ii'. PVi'} _SK) =|>
             State':=2 /\ secret ({IDi, PWi}, s1, EDi)
                    /\ RS':= new ()
                    /\ CK':= H (RS'. XCS . ET . Ii')
                    /\ CK1':= E (CK'. P)
                    /\ secret ({XCS, ET}, s2, CS)
                    /\ SND ({CK1'} _SK)
    % Login and Authentication phase
            2. State= 0/\ RCV (P1'. P2'. EIi') =|>
             State':= 4 /\ R2':=new ()
                    /\ RS':=new ()
                    /\ CK':=new ()
                    /\ P3':= E (R2'. P)
                    /\ Ai':= H (xor(xor(RS', H(XCS)), CK'))
                    /\ Ai1':= E (Ai'. P)
                    /\ P4':= H (P1'. E (R2'. Ai1'))
                    /\ KPV':= xor(PVx, PVy)
                    /\ Ti':= xor(RS, H(XCS))
                    /\ Ti1':= xor(Ti', KPV')
                    /\ SND (P3'.P4 .Ti1')
                    /\ witness (CS, EDi, cs_ed_r2, R2')

            3. State= 4/\ RCV ({Vi'} _SK) =|>
             State':=6 /\ R1':=new ()
                    /\ R2':=new ()
                    /\ P1':=new ()
                    /\ SK':= E (R1'. E (R2'. E (PWi . P)))
                    /\ Vi':= H (SK'. E (R2'. P1'))
                    /\ request (EDi, CS, ed_cs_r1, R1')
end role
```

and verifies $P_4^* \stackrel{?}{=} P_4$ to authenticate cloud server. By observing this process, it is found that above conditions are satisfied. Hence, it is concluded that the proposed protocol provides proper mutual authentication. In contrast, in the existing protocols [31, 35], the embedded device cannot compute $A_i = H(T_i \oplus PW_i \oplus CK')$ and $A_i = H(B_i \oplus CK' \oplus H(S\_ID_i|PW_i))$ since it does not have the knowledge of PW_i and B_i. Hence, the verification $P_4^* \stackrel{?}{=} P_4$ is not possible. Thus, the protocols [31, 35] failed to provide the mutual authentication.

5.3.2 S2: replay attack

In the proposed protocol, an adversary may try to capture the transmission message { P_1,P_2,I_i } which is transmitted from device to server. The adversary may login as a legal device by re-transmitting the captured message to affect the replay attack. After receiving the login request, the server will assume that replay attack has been occurred as the status bit is already set to ‘1’ for the previously logged device. If it is assumed that by any means adversary impersonates the legal device, then, after receiving adversary login request, CS retrieves the data associated to I_i and computes CK and P_2^*. Afterward, CS verifies $P_2^* \stackrel{?}{=} P_2$ and delivers {P_3,P_4, T_i'} to ED_i. However, upon receiving the message {P_3, P_4, T_i'}, the adversary is unable to calculate $A_i = H(T_i \oplus CK')$ without the knowledge of T_i because of the encrypted T_i where, $T_i' = T_i \oplus K_{PV}$ is sent through the channel and K_{PV} is only computed by ED_i and CS. Moreover, it will not be easy for the adversary to calculate the session key $SK = R_1 \cdot PW_i \cdot P_3 = R_1 \cdot R_2 \cdot PW_i \cdot P$ and the authentication parameter $V_i = H(SK \parallel R_1 \cdot PW_i \cdot P_3)$. Thus, the proposed protocol is free from replay attack.

Fig. 4 HLPSL code for role specification of session, goal and environment

```
role session (EDi, CS: agent, SK: symmetric_key, H:hash_func)
def=
      local SE, RE, SC, RC: channel(dy)
      composition
            embedded_device (EDi, CS, SK, H, SE, RE)
         /\ cloud_server (EDi, CS, SK, H, SC, RC)
end role
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
role environment()
def=
      const edi, cs:agent,
            sk: symmetric_key,
            f: hash_func,
            p1, p2, p3, p4, ti, eii, vi: text,
            s1, s2, ed_cs_r1, cs_ed_r2: protocol_id
       intruder_knowledge = {edi, cs, f, p1, p2, eii, p3, p4, ti, vi}

composition
      session(edi, cs, sk, f)
   /\ session (i, cs, sk, f)
   /\ session (edi, i, sk, f)
end role
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
goal
      secrecy_of s1
      secrecy_of s2
      authentication_on ed_cs_r1
      authentication_on cs_ed_r2
end goal
environment ()
```

Fig. 5 The simulation results of the proposed protocol using OFMC and CL-AtSe back-ends

```
% OFMC
% Version of 2006/02/13
SUMMARY
 SAFE
DETAILS
 BOUNDED_NUMBER_OF_SESSIONS
PROTOCOL
 /home/span/span/testsuite/results/Proposed_IoT.if
GOAL
 as_specified
BACKEND
 OFMC
COMMENTS
STATISTICS
 parseTime: 0.00s
 searchTime: 0.26s
 visitedNodes: 244 nodes
 depth: 4 plies
```

```
SUMMARY
 SAFE

DETAILS
 BOUNDED_NUMBER_OF_SESSIONS
 TYPED_MODEL

PROTOCOL
 /home/span/span/testsuite/results/Proposed_IoT.if

GOAL
 As Specified

BACKEND
 CL-AtSe

STATISTICS

 Analysed: 0 states
 Reachable: 0 states
 Translation: 0.01 seconds
 Computation: 0.00 seconds
```

5.3.3 S3: password guessing attack

The password guessing attack is a vital problem in any password based secure authentication scheme. In the proposed protocol, the password verifier $PV_i = PW_i \cdot P$ is stored in the server in a write protected file and it is difficult for the adversary to retrieve the password PW_i from PV_i due to the hard of ECDLP. Hence, the password guessing attack is not possible in the proposed protocol. In contrast, in the existing protocol [24], the adversary retrieves a password PW_i in a following manner;

Assume that ED_i's password is PW_1 and it is used to calculate $A_i^* = H(T_i \oplus PW_1 \oplus CK')$. Next, $P_4^* = P_3 \cdot A_i^*$ is computed and the condition $P_4^* \stackrel{?}{=} P_4$ is verified to find the correct value of PW_1. If the condition is satisfied, the adversary will consider $PW_i = PW_1$. Otherwise, the process will continue till the adversary obtains the proper password PW_i. Similar process can be followed for the protocol [32] to obtain the

Table 4 Security comparison of the proposed protocol with other existing protocols

Reference protocols	Security attributes									
	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10
Hafizul et al. [10]	Yes	Yes	–	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Liao et al. [13]	No	Yes	–	Yes	–	No	No	Yes	Yes	Yes
Kalra et al. [31]	No	Yes	No	No	No	No	No	No	No	No
Chang et al. [32]	Yes	Yes	No	No	No	Yes	No	No	Yes	Yes
Wang et al. [33]	Yes	Yes	–	No	–	Yes	No	No	Yes	Yes
Kumari et al. [34]	Yes	Yes	Yes	Yes	Yes	Yes	No	No	Yes	Yes
Bhubaneswari et al. [35]	No	Yes	Yes	Yes	Yes	No	No	No	No	No
Proposed protocol	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes

Yes: prevents the attack or supports a specific attribute; No: unable to prevent the attack or does not support an attribute; —: not applicable in a protocol

password PW_i. Hence, these protocols can get affected by password guessing attack.

5.3.4 S4: device privacy

To ensure the privacy of the device, the identity of the device should not be transmitted directly without protection. In the login and authentication phase of the proposed protocol, device transmits $\{P_1, P_2, EI_i\}$ to the server. Here, EI_i is the encryption version of protected identity I_i i.e. $EI_i = I_i \oplus K_{PV}$, where, $K_{PV} = PV_x \oplus PV_y$. Moreover, K_{PV} is calculated from PV which is difficult to calculate by an adversary due to the fact that PV is never transmitted through any messages in the login and authentication phase. Therefore, the proposed protocol preserves the device privacy. However, in the existing protocols [31, 33], the identity of the device ID_i is transmitted directly from ED_i to CS through the login request message $\{P_1, P_2, ID_i\}$ during login and authentication phase. Thus, these protocols fail to preserve device privacy.

5.3.5 S5: insider attack

Insider attack can occur when a privileged insider steals the password from the server's information to use it for accessing other servers (where the device is previously registered with the same information) by making a login request. In the proposed protocol, a password verifier table has been maintained which contains protected device identity I_i, password verifier $PV_i = PW_i \cdot P$ and a status bit. The retrieval of the password PW_i from the password verifier PV_i is impossible due to the hard of ECDLP. Hence, the proposed protocol prevents the insider attack. In the existing protocols [31, 32], password PW_i is generated by CS for every ED_i during the registration phase. Consequently, the insider of CS easily gets the password PW_i which can be misused. Hence, the protocols [31, 32] are vulnerable to insider attack.

5.3.6 S6: man-in-the-middle attack

In the proposed protocol, due to the achievement of mutual authentication between ED_i and CS, man-in-the-middle attack is not feasible. However, the existing protocols [13, 14, 35] do not achieve mutual authentication. Thus, man-in-the-middle attack is feasible for the existing protocols [13, 14, 35].

5.3.7 S7: impersonation attack

If the adversary accesses the security parameters stored in the server, the impersonate attack takes place. In the proposed protocol, the server stores $\{PV_i,\ ET_i = T_i \oplus X_{CS},\ EA_i' = A_i' \oplus R_S$ and $EE_T = E_T \oplus R_S\}$ corresponding to I_i of the device ED_i in its database. Let us assume that the server compromises the stored value. Under this situation also the adversary cannot access the values of $\{T_i, A_i', E_T\}$ because these are protected by the random nonce R_S and the secret key X_{CS} of the cloud server and PV_i is stored with a status bit in a write protected mode. Moreover, without knowing the value of $\{T_i, A_i', E_T\}$, R_S and X_{CS}, it is impossible to obtains the cookie $CK = H(R_S \parallel X_{CS} \parallel E_T \parallel I_i)$ to validate the login request. Furthermore, it is not possible to communicate further for authentication. Therefore, the proposed protocol is immune to impersonation attack. In contrast, in the existing protocol [31], during registration process, the cloud server stores $\{A_i', T_i, ID_i$ and $E_T\}$ in its database. If the server compromises these values, the adversary can impersonate as server as follows: In the login and authentication process, the adversary intercepts the login request message $\{P_1, P_2, ID_i\}$. Then, it retrieves the values associated with ID_i from the captured values. Afterward, the adversary selects a random number R_x, computes $P_{3x} = R_x \cdot P$ and $P_{4x} = R_x \cdot A_i'$ and subsequently sends $\{P_{3x}, P_{4x}, T_i\}$ to ED_i. Upon receiving $\{P_{3x}, P_{4x}, T_i\}$, device ED_i would calculate $P_{4x}^* = A_i \cdot P_{3x}$. Since, $P_{4x}^* = A_i \cdot P_{3x} = A_i \cdot R_x \cdot P = R_x \cdot A_i' = P_{4x}$, device will confirm that it is connected to the legal server. Thus, it is easy for the adversary to impersonate as server. By following the similar process, it can be said that the impersonation attack is feasible for the protocols [32–35].

5.3.8 S8: many logged-in device's attack

If the identity and password of the legal devices are exposed by any means to many adversaries, then, by using that information the adversaries can access the account of the legal device resulting in many logged-in devices' attack. In the proposed system, many adversaries can try to access the account by using the proper identity and password of the legal device but only a single adversary can access the account. This is due to the fact that when a device logs in, the status bit is set to '1.' In the meantime, if other adversaries use the same information to log into the server, then the server rejects the attempt because the status bit indicates that some device is already logged in. Hence, the proposed protocol is free from many logged-in devices' attack. However, for the protocols [10, 31–35], if the identity and password are leaked, they are unable to prevent the many logged-in devices' attack as they have not included the concept of setting the login status of the logged device.

5.3.9 S9: session key agreement

In the proposed protocol, during the authentication process, the device and the server individually generates the session key $SK = R_1 \cdot PW_i \cdot P_3 = R_2 \cdot P_1 = R_1 \cdot R_2 \cdot PW_i \cdot P$ and shares it. Since the computation of the session key depends on the device password PW_i and the random nonce R_1 and R_2, it is impossible for the adversary to compute the session key. Thus, the session key agreement is achieved properly. However, in the existing protocol [31] the computation of session key $SK = H(X_{CS} \parallel ID_i \parallel R_1 \parallel R_2)$ is not possible due to the fact that neither ED_i has the knowledge of R_2 and X_{CS} nor CS has the knowledge of R_1. Correspondingly, in the protocol [35], the verification $V_i^* \stackrel{?}{=} V_i$ is false and hence session key cannot be generated. Thus, the session key agreement is not feasible in the protocols [31, 35].

5.3.10 S10: perfect forward secrecy

Perfect forward secrecy indicates that the session keys should not be affected by the adversary even if the device's password PW_i and the cloud server's secret key X_{CS} are exposed. In the proposed protocol, the session key $SK = R_1 \cdot PW_i \cdot P_3 = R_2 \cdot P_1 = R_1 \cdot R_2 \cdot PW_i \cdot P$ has been generated by the device and the server individually. Assuming the adversary has the knowledge of PW_i and X_{CS}, it is impossible to generate the session key because it requires random nonce R_1 and R_2. If the adversary tries to retrieve R_1 and R_2 from the pair $(P_1, P_2) = (R_1 \cdot PW_i \cdot P, R_2 \cdot P)$, it is difficult to find due to the hard of CDHP. Therefore, the proposed protocol achieves perfect forward secrecy. In contrast, the protocols [31, 35] cannot achieve perfect forward secrecy because session key agreement is not feasible as mentioned in S8.

6 Performance analysis

In this section, the performance of the proposed protocol has been analyzed and compared with the existing related protocols [10, 13, 31–35] with respect to computational overhead, bandwidth consumption, storage overhead and total computational time.

6.1 Computational overhead

A comparison of the computational overhead of the proposed protocol with the existing related protocols is presented in Table 5. Since, in an authentication protocol, the login and authentication phase is executed more frequently as compared to other phases, only this phase has been considered for the purpose of calculation. In this regard, T_H, T_{EPM} and T_{ECA} have been denoted as the computational time of hash operation, elliptic curve point multiplication and elliptic curve point addition, respectively. During calculation, the computational overhead of some lightweight operations such as XOR, concatenation, comparison, etc., have been ignored because of their insignificant impact as compared to other operations.

From Table 5, it is found that the computational overhead of the proposed protocol is lesser than the related protocols [10, 32, 33]. However, the computational overhead of the proposed protocol is little higher than the protocols [13, 31, 34, 35]. This is due to the fact that, the proposed protocol achieves forward secrecy through the session key agreement between embedded device and cloud server which is not feasible in the protocol [31, 35]. Moreover, the proposed protocol adopts password verifier and uses more security function to avoid some of the security flaws which are cannot prevent by the protocol [13].

6.2 Communication overhead

Bandwidth consumption is the essential measure of communication overhead. Bandwidth consumption of the proposed protocol is equivalent to the total size of the login and authentication messages. For calculating the size of the login and authentication messages, the length of following parameters has been assumed:

Table 5 Computational overhead of the proposed protocol and related protocols

Protocols	Login and authentication phase		Total
	Embedded device	Cloud server	
Hafizul et al. [10]	$3T_H + 2T_{ECA} + 3T_{EPM}$	$3T_H + 2T_{ECA} + 4T_{EPM}$	$6T_H + 4T_{ECA} + 7T_{EPM}$
Liao et al. [13]	$2T_{ECA} + 3T_{EPM}$	$5T_{ECA} + 3T_{EPM}$	$7T_{ECA} + 6T_{EPM}$
Kalra et al. [31]	$4T_H + 3T_{EPM}$	$5T_H + 4T_{EPM}$	$9T_H + 7T_{EPM}$
Chang et al. [32]	$5T_H + 4T_{EPM}$	$5T_H + 4T_{EPM}$	$10T_H + 8T_{EPM}$
Wang et al. [33]	$5T_H + 4T_{EPM}$	$6T_H + 4T_{EPM}$	$11T_H + 8T_{EPM}$
Kumari et al. [34]	$3T_H + 4T_{EPM}$	$4T_H + 4T_{EPM}$	$7T_H + 8T_{EPM}$
Bhubaneswari et al. [35]	$3T_H + 4T_{EPM}$	$5T_H + 4T_{EPM}$	$8T_H + 8T_{EPM}$
Proposed protocol	$4T_H + 4T_{EPM}$	$5T_H + 4T_{EPM}$	$9T_H + 8T_{EPM}$

- The length of the each of the random nonce (R_1, R_2, R_s) is 160 bits.
- The length of device identity ID_i is 160 bits.
- The length of the each of the security parameters $\{CK', T_i', V_i\}$ is 160 bits.
- The length of the output of hash function (SHA-1) [44] is 160 bits.
- Since the security strength of 160 bit ECC is equivalent to 1024 bit RSA cryptosystem [37, 45], an ECC point $P = (P_x, P_y)$ needs $(160 + 160) = 320$ bits [46].

The calculation of the size of the login and authentication messages of the proposed protocol has been analyzed bellow:

Message $1 = P_1 \parallel P_2 \parallel EI_i = 320 + 320 + 160 = 800$ bits

Message $2 = P_3 \parallel P_4 \parallel T_i' = 320 + 320 + 160 = 800$ bits

Message $3 = V_i = 160$ bits

Therefore, bandwidth consumption of the proposed protocol is:

Bandwidth $= \sum_{i=1}^{3} Message(i) = 1760$ bits.

The bandwidth consumption of the proposed protocol and the related protocols [10, 13, 31–35] is presented in Table 6.

Table 6 shows that the bandwidth consumption of the proposed protocol is the same as the related protocols [31–35] and little larger than the protocols [10, 13]. Hence, the proposed protocol has equivalent communication overhead as compared to the protocols [31–35] and competitive value with the protocols [10, 13].

Table 6 Bandwidth consumption of the proposed protocol and related protocols

Protocols	Bandwidth consumption	
	Number of messages	Number of bits
Hafizul et al. [10]	3	1440
Liao et al. [13]	3	1280
Kalra et al. [31]	3	1760
Chang et al. [32]	3	1760
Wang et al. [33]	3	1760
Kumari et al. [34]	3	1760
Bhubaneswari et al. [35]	3	1760
Proposed protocol	3	1760

6.3 Storage overhead

In this section, the storage overhead of the proposed protocol and some related protocols has been presented and compared. Here, the storage overhead of the embedded device has been considered for the purpose of calculation since it has minute memory as compared to the server memory. In the proposed protocol, the cookie CK' is stored in the embedded device (ED_i). The memory required by the ED_i to store the cookie CK' is 320 bits. Similarly, in the protocols [31, 33] the ED_i stores $CK' = 320$ bits in its memory. However, in the protocol [32], the ED_i stores $\{CK', H(PW_i)\} = 320 + 160 = 480$ bits in its memory. Correspondingly, in the protocols [10] and [13], the device stores 1120 bits and 480 bits, respectively [31]. Comparison of the storage overhead of the embedded device of the proposed protocol with respect to the related protocols is illustrated in Table 7 and Fig. 6.

Figure 6 shows that the storage overhead of the embedded device in the proposed protocol is equivalent to the storage overhead of the protocols [31, 33, 35]. Moreover, the memory required by the embedded device in the proposed protocol is much lesser than the protocols [10, 13, 32, 34].

6.4 Computational time

The total computational time of the proposed protocol and the other related protocols is presented in Fig. 7. Here, the simulation has been performed by using MATLAB 2015a environment.

From Fig. 7 it is noticed that the computational time of the proposed protocol is little larger than the protocols [31–35] which consumes 465.39 s. This is due to the fact that the proposed protocol adopts the concept of password verifier

Table 7 Comparison of storage overhead of the proposed protocol with related protocols

Protocols	Storage overhead (bits)
Hafizul et al. [10]	480
Liao et al. [13]	1120
Kalra et al. [31]	320
Chang et al. [32]	480
Wang et al. [33]	320
Kumari et al. [34]	480
Bhubaneswari et al. [35]	320
Proposed protocol	320

with the status bit and also uses some more security parameters to protect the system from various security attacks such as impersonation attack, device privacy, password guessing attack, insider attack, many login device's attack and provide perfect forward secrecy as well as achieves proper mutual authentication which the protocols [31–35] cannot prevent. Hence, it can be said that the proposed protocol achieves greater security than the protocols [31–35] with the competitive computational time.

6.5 Discussion

The overall outcomes of the above analysis have been summarized below:

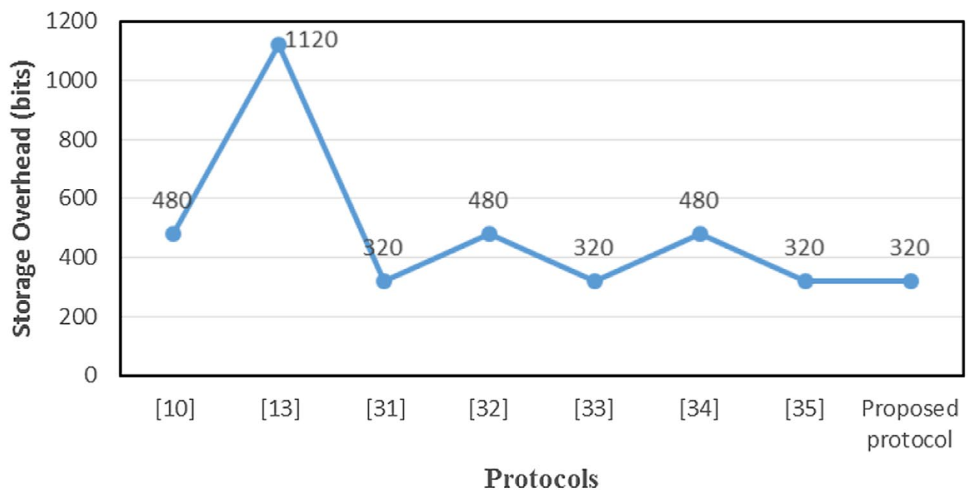

Fig. 6 Comparative storage overhead

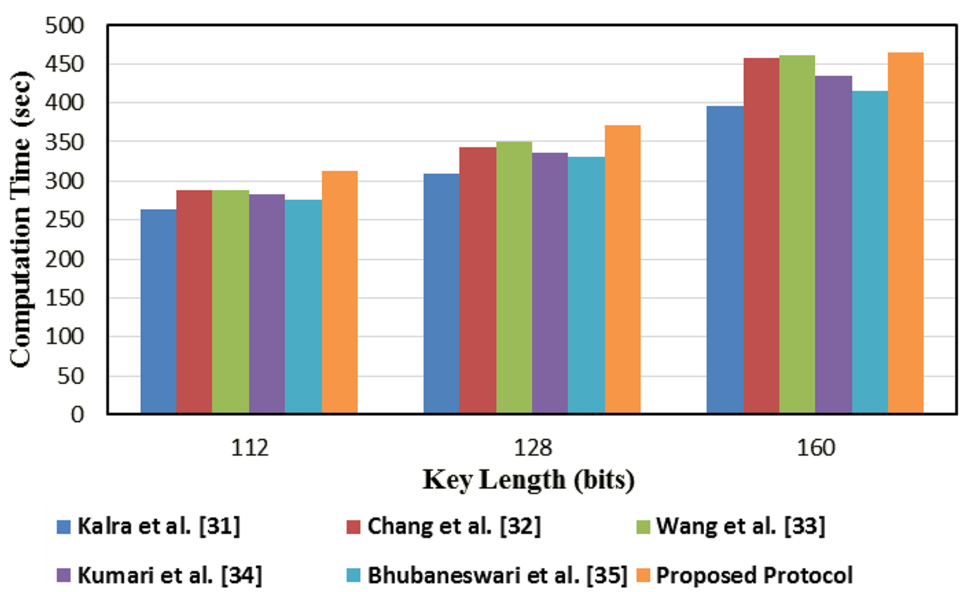

Fig. 7 Total computational time of the proposed protocol and the related protocols

1. The proposed protocol achieves mutual authentication where the protocol [31] does not.
2. The proposed protocol attains better security than the related protocols [10, 13, 31–35].
3. The proposed protocol outperforms the protocols [10, 32, 33] in terms of computational overhead. The proposed protocol is also superior to the protocols [10, 13, 32, 34] as far as the storage overhead is concerned. However, the computational overhead of the proposed protocol is little higher than the protocols [13, 31, 34, 35] because the proposed protocol attains forward secrecy through the session key agreement between ED_i and CS which is not feasible in the protocols [31, 35] and also achieves better security than the protocols [13, 31, 34, 35].
4. The proposed protocol consumes little more time than the related existing protocols [31–35] for the total computation. The reason is that the proposed protocol employs a password verifier and some additional security parameters to defend several attacks which the protocols [31–35] are unable to prevent.
5. Overall, our proposed protocol outperforms the related protocols [10, 13, 31–35] in all respect.

7 Conclusions and future work

In this work, an ECC-based mutual authentication and security protocol has been proposed for the IoT and cloud servers. Earlier related existing authentication protocols for the IoT and cloud servers failed to provide the necessary security requirements as required. Simulation for the formal security analysis of the proposed protocol using AVISPA tool ensures that the protocol is safe and secure from various security attacks. Moreover, the informal security analysis of the present work shows that the proposed protocol attains higher security than the related protocols [10, 13, 31–35]. The performance analysis of the present work finds that the computational overhead of the proposed protocol is lesser than the protocols [10, 32, 33]. Furthermore, the communication and storage overhead of the proposed protocol is equivalent to the protocols [31–35] and [31, 33, 35], respectively, and also needs much lesser storage overhead than the protocols [10, 13, 32, 34]. However, the total computational time and the computational overhead of the proposed protocol are little larger than the protocols [31–35] and [13, 31, 34, 35], respectively. Hence, it can be concluded that our proposed protocol is capable enough to provide an improved secure mutual authentication model for IoT and cloud server environments.

In the future, our work can be extended toward the further improvement of the total computational time and the computational overhead of the proposed protocol without sacrificing the level of security. We would also like to derive the behavior and reliability model for the proposed protocol so that the users could have prior knowledge about the system behaviors and reliabilities before using the model. The proposed protocol can be applicable to any IoT industries, where the data security and the authentication are the prime important part of the integration of embedded devices and cloud servers.

References

1. Atzori L, Lera A, Morabito G (2010) The Internet of Things: a survey. Comput Netw 54:2787–2805
2. Al-Fuqaha A, Guizani M, Mohammadi M, Aledhari M, Ayyash M (2015) Internet of Things: a survey on enabling technologies, protocols, and applications. IEEE Commun Surv Tutor 17(4):2347–2376
3. Kouicem DE, Bouabdallah A, Lakhlef H (2018) Internet of Things security: a top-down survey. Comput Netw 141:199–221
4. Botta A, Donato WD, Persico V, Pescape A (2016) Integration of cloud computing and Internet of things: a survey. Future Gener Comput Syst 56:684–700
5. Sascha M, Sebastian W (2008) Secure communication in microcomputer bus systems for embedded devices. J Syst Archit 54:1065–1076
6. Debiao H, Sherali Z (2015) An analysis of RFID authentication schemes for Internet of Things in healthcare environment using elliptic curve cryptography. IEEE Internet Things J 2(1):72–83
7. Afreen R, Mehrotra SC (2011) A review on elliptic curve cryptography for embedded systems. J Comput Sci Inf Technol 3(3):84–103
8. Yang J, Chang C (2009) An ID-based remote mutual authentication with key agreement protocol for on elliptic curve cryptosystem. Comput Secur 28:138–143
9. Yoon EJ, Yoo KY (2009) Robust ID-based remote mutual authentication with key agreement protocol for mobile devices on ECC. In: Proceedings of the international conference on computational science and engineering, pp 633–640
10. Hafizul SK, Biswas GP (2011) A more efficient and secure ID-based remote mutual authentication with key agreement scheme for mobile devices on elliptic curve crypto systems. J Syst Softw 84(11):1892–1898
11. Chou CH, Tsai KY, Lu CF (2013) Two ID-based authenticated schemes with key agreement for mobile environments. J Supercomput 66(2):973–988
12. Farash MS, Attari MA (2014) A secure and efficient identity-based authenticated key exchange protocol for mobile client–server networks. J Supercomput 69:395–411
13. Liao YP, Hsiao CM (2014) A secure ECC-based RFID authentication scheme integrated with ID-verifier transfer protocol. Ad Hoc Netw 18:133–146
14. Peeters R, Hermans J (2013) Attack on Liao and Hsiao's Secure ECC based RFID authentication scheme integrated with ID-verifier transfer protocol. Cryptology ePrint Archive. Report 2013/399
15. Moosavi SR, Nigussie E, Virtanen S, Isoaho J (2014) An elliptic curve-based mutual authentication scheme for RFID implants systems. Procedia Comput Sci 32:198–206
16. Khatwani C, Roy S (2015) Security analysis of ECC based authentication protocols. In: Proceedings of ieee international conference on computational intelligence and communication networks, pp 1167–1172

17. Abbasinezhad-Mood D, Nikooghadam M (2018) Efficient design of a novel ECC-based public key scheme for medical data protection by utilization of NanoPi fire. IEEE Trans Reliab 67(3):1328–1339
18. Abbasinezhad-Mood D, Nikooghadam M (2018) Efficient anonymous password-authenticated key exchange protocol to read isolated smart meters by utilization of extended chebyshev chaotic maps. IEEE Trans Ind Inf 4(11):4815–4828
19. Abbasinezhad-Mood D, Ostad-Sharif A, Nikooghadam M (2019) Novel anonymous key establishment protocol for isolated smart meters. IEEE Trans Ind Electron 67(4):2844–2851
20. Alshahrani M, Traore I (2019) Secure mutual authentication and automated access control for IoT smart home using cumulative Keyed-hash chain. J Inf Secur Appl 45:156–175
21. Li X, Niu J, Bhuiyan MZA, Wu F, Karuppiah M, Kumari S (2018) A robust ECC based provable secure authentication protocol with privacy preserving for Industrial Internet of Things. IEEE Trans Ind Inf 14(8):3599–3609
22. Alcaide A, Palomar E, Montero-Castillo J, Ribagorda A (2013) Anonymous authentication for privacy-preserving IoT target-driven applications. Comput Secur 37:111–123
23. Lin X-J, Sun L, Qu H (2015) Insecurity of an anonymous authentication for privacy-preserving IoT target-driven applications. Comput Secur 48:142–149
24. Dhillon PK, Kalra S (2017) Secure multi-factor remote user authentication scheme for Internet of Things environments. Int J Commun Syst 6:e3323
25. Ostad-Sharif A, Arshad H, Nikooghadam M, Abbasinezhad-Mood D (2019) Three party secure data transmission in IoT networks through design of a lightweight authenticated key agreement scheme. Future Gener Comput Syst 100:82–892
26. Waquar A, Raza A, Abbas H, Khan MK (2013) A framework for preservation of cloud users' data privacy using dynamic reconstruction of metadata. J Netw Comput Appl 36:235–248
27. Distefano S, Merlino G, Puliafito A (2015) A utility paradigm for IoT: the sensing cloud. Pervasive Mob Comput 20:127–144
28. Persson P, Angelsmark O (2015) Calvin—merging cloud and IoT. Procedia Comput Sci 52:210–217
29. Stergiou C, Psannis KE, Kim B-G, Gupta B (2018) Secure integration of IoT and cloud computing. Future Gener Comput Syst 78:964–975
30. Chatterjee S, Samaddar SG (2020) A robust lightweight ECC-based three-way authentication scheme for IoT in cloud. In: Elçi A, Sa P, Modi C, Olague G, Sahoo M, Bakshi S (eds) Smart computing paradigms: new progresses and challenges Advances in intelligent systems and computing, vol 767. Springer, Singapore
31. Kalra S, Sood SK (2015) Secure authentication scheme for IOT and cloud servers. Pervasive Mob Comput 24:210–223
32. Chang C-C, Wu H-L, Sun C-Y (2017) Notes on secure authentication scheme for IOT and cloud servers. Pervasive Mob Comput 38:275–278
33. Wang K-H, Chen C-M, Fang W, Wu T-Y (2017) A secure authentication scheme for internet of things. Pervasive Mob Comput 42:15–26
34. Kumari S, Karuppiah M, Das AK (2018) A secure authentication scheme based on elliptic curve cryptography for IoT and cloud servers. J Supercomput 74:6428–6453
35. Bhubaneswari S, Ananth NV (2018) Enhanced mutual authentication scheme for cloud of things. Int J Pure Appl Math 119(15):1571–1583
36. Hankerson D, Menezes A, Vanstone S (2004) Guide to elliptic curve cryptography. Springer, New York
37. Mahto D, Khan DA, Yadav DK (2016) Security analysis of elliptic curve cryptography and RSA. In: Proceedings of the world congress on engineering, pp 1–4
38. Wu F, Xu L, Kumari S, Li X (2018) An improved and provably secure three-factor user authentication scheme for wireless sensor networks. Peer-to-Peer Netw Appl 11(1):1–20
39. Panda PK, Chattopadhyay S (2019) An improved authentication and security scheme for LTE/LTE-a networks. J Ambient Intell Hum Comput. https://doi.org/10.1007/s12652-019-01248-8
40. Vigano L (2006) Automated security protocol analysis with the AVISPA tool. Electron Notes Theor Comput Sci 155:61–86
41. [Online]. AVISPA: automated validation of internet security protocols and applications. Accessed Jan (2018). http://www.avispaproject.org/
42. Wazid M, Das AK, Odelu V, Kumar N, Conti M, Jo M (2018) Design of secure user authenticated key management protocol for generic IoT networks. IEEE Internet Things J 5(1):269–282
43. Dolev D, Yao AC (1983) On the security of public key protocols. IEEE Trans Inf Theory 29(2):198–208
44. Secure hash standard (1995) Nat. Inst. Standards Technol. (NIST), USA, Tech. Rep. FIPS PUB: 180-1
45. Panda PK, Chattopadhyay S (2019) A modified PKM environment for the security enhancement of IEEE 802.16e. Comput Standard Interface 61:107–120
46. Challa S, Wazid M, Das AK, Kumar N, Reddy AG, Yoon E-J, Yoo K-Y (2017) Secure signature based authenticated key establishment scheme for future IOT applications. IEEE Access 5:3028–3043